Presentation Script

Topics : Compiler Design , Computation

Group : **Nopa Islam & Sukannya Saha**

Slide : 1

Good After- Noon everyone ,welcome to our group presentation. Our tentative topics is Compiler Design , Computation

Slide : 2

I am Nopa Islam staring the presentation , with my fellow team mate Sukannya Saha. So Lets log in to the presentation.

Slide : 3

Compiler is mainly one kind of program that normally converts instructions into machine code.

Normally it handle 5 types of code , they are

Source Code => It is mainly the collection of instructions

Object Code => It means sequence of statements

Byte Code => Executed by parsing and directly executing the instructions

Machine Code => Executed by CPU

Micro Code => Lowest specified level of processor and machine instructions set

Slide : 4

Let's look at the flow , so compiler takes a source program as input ,generates all possible error reports ,if not then takes the input in the .exe and produces result.

Slide : 5

Here we can see, compiler is working on 6 steps. Each of them are well defined. The front end is well maintained. The backend does all the calculation

Slide : 6

So preprocessor modifies the input, mainly deletes all spaces . Compiler does all the critical operation in 2 parts they are (Analysis & Synthesis). Assembler just converts the low level code to machine code. Linker + loader supplies relocate –able object code.

Memory holds the output.

My part is done now ; I wanna call My Team Mate Sukannya Saha to continue the presentation. Over to you Sukannya

Thanks Nopa for giving me the scope.

So I will start from Slide number 7

Slide : 7

So Nopa showed us the compiler compilation steps. Here we can see the whole process nicely. Starting from the lexical analysis part to machine independent code optimization the flow is connected with a data structure called “Symbol-Table”

Using Grammar to make tokens and then install it in the symbol table. All phases are connected with the table.

Slide : 8

So lexical analysis part does the mechanism to make tokens.

Syntax analysis part generates hierarchy structure , which is called parse tree. Semantic analysis part does the type checking part ,which is the last part of error detection. Intermediate code generation does 3 address code, it is a sort of precise expression. Optimization mainly depends on the designer. Code generator maps the intermediate code with the input

Slide : 9

Here we can see , how compiler evaluates an expression in every step . Which depends on symbol table . The flow is sequential .

Slide : 10

We are designing a compiler, so in our computation , error can occur . Normally 4 types of error can be found .

Lexical Error : If we incorrectly type the id name, it can be a lexical error. Most of the time , compiler can not find it out

Syntactical Error : ; , () , {} ,[] if we miss these things

Semantical Error : Incompatible type like int a = “Chill ”; is one kind of semantical error

Logical Error : Most common error is this header miss, infinite loop, it can crash the code

Slide : 11

In order to recover from error we can follow these strategy

Panic mode : It follows shortest amount change to run the code correctly

Statement Mode : It takes corrective measures parse ahead. It does prediction .

Error Productions : Creating augmented grammar.

Global Corrections : It finds the closest match .

Abstract Syntax Tree : It is a special kind of data structures . Having the abstract syntactic structure of source code written in a programming language . Compared to the source code, an AST does not include inessential punctuation and delimiters .

Slide : 12

Any question from the audience ?

Slide : 13

So it’s time to log out from the presentation. Thanks my team mate and all for your Cooperation. Have a good day .